Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Principios de sistemas operativos

Tarea 2

Daemons

Estudiante	**Carné**
Rodríguez Rojas Andrés	2019279722

Profesor:

Leaonardo Araya Martínez

28 de mayo de 2025

I. Introducción

En este trabajo enfatiza en el desarrollo y uso de procesos en segundo plano o "daemons" con el objetivo de llevar a cabo tareas específicas que no requieran de ningún tipo de interacción por parte del usuario final. Se establece el objetivo de implementar un monitor de temperatura que realice registros de las temperaturas del CPU del equipo en uso, con el objetivo de detectar aumentos de temperatura y notificar casos en los que se alcance una temperatura que pueda representar un riesgo para el equipo.

II. Ambiente de desarrollo

El desarrollo de este proyecto requirió la utilización de herramientas y bibliotecas específicas para lograr la implementación de un "daemon" funcional en un sistema operativo Linux. El ambiente de desarrollo se centró en el uso del lenguaje Python 3, aprovechando su versatilidad y la disponibilidad de módulos estándar que facilitan la interacción con el sistema operativo, no obstante, el proyecto no requirió frameworks complejos ni aplicaciones de terceros más allá de las bibliotecas estándar de Python.

II-A. Biblioteca: `subprocess`

La biblioteca `subprocess` fue fundamental en este proyecto, permitiendo la ejecución de comandos del sistema directamente desde el código Python. Esta biblioteca fue utilizada principalmente para dos propósitos críticos, el primero de ellos fue la obtención de la temperatura del CPU a través del comando `sensors`, propio de la librería `im-sensors`, de modo que haciendo uso de la función `subprocess.check_output()` permitió guardar la temperatura del sistema y utilizarla para las comprobaciones necesarias.

Por otra parte, la biblioteca `subprocess` también permitió el envío de notificaciones de sistema a partir de su método `subprocess.run()`.

II-B. Biblioteca: `signal`

Por parte de la biblioteca `signal`, la constante $SIGTERM$ fue empleada para detener el daemon de manera controlada cuando se recibía el comando de detención, lo que permitió una terminación limpia, cerrando adecuadamente los archivos abiertos y eliminando el archivo PID antes de finalizar.

II-C. Paquete `im-sensors`

Este paquete, aunque no es una biblioteca Python, fue esencial para el funcionamiento del sistema, ya que proporcionó los controladores e instrucciones necesarias para monitorear temperaturas del CPU de forma sencilla y certera.

III. Atributos

III-A. Necesidades de aprendizaje

Para el desarrollo de este proyecto fue necesario un breve proceso de investigación relacionado con los procesos en segundo plano y la gestión de los mismos, ya que a la hora de crear el daemon es necesario que dicho recurso sea creado como un proceso independiente de interacciones o terminales, de modo que pueda seguirse ejecutando en segundo plano en todo momento. Así se identifica una de las necesidades de aprendizaje de este proceso de desarrollo en la creación y comportamiento de procesos en segundo plano, de modo que estos lleven a cabo la tarea planteada y sean permanentemente independientes del usuario final del programa.

Además, otro de los desafíos encontrados es la monitorización de la temperatura del procesador, ya que se requieren herramientas y librerías externas para dicho fin, por lo tanto es necesaria la búsqueda e implementación de alternativas que permitan obtener de manera acertada la temperatura interna del sistema y su pertinente análisis y utilización.

III-B. Tecnologías de desarrollo

Para el desarrollo de este proyecto no fue necesario el uso de tecnologías complejas, por lo cual, además del uso de lenguajes de programación, sus correspondientes y posibles ambientes de desarrollo y las librerías que pueden incluir los mismos, no fue necesario el uso de software o equipo adicional.

III-C. Planificación del desarrollo

En términos de planificación y estrategias de trabajo, fue necesario llevar a cabo un proceso investigativo previo, debido a la existencia de incertidumbre o desconocimiento en cuanto a los aspectos detallados en la sección de necesidades de aprendizaje (III-A) de este documento. Asimismo, resultó útil la búsqueda de información textual y el uso de herramientas como YouTube, StackOverflow, entre otras herramientas que permitieron una mayor comprensión y manejo del objetivo a evaluar e implementar en este proyecto.

Posteriormente, al iniciar el proceso de desarrollo, la información obtenida permitió implementar la solución final de un modo fluido y comprensible, de modo que, debido al buen proceso de análisis e investigación realizado, se facilitó la estructuración y creación de la solución implementada.

III-D. Evaluación del proceso de planificación

Como se hace mención en la sección de planificación del desarrollo (III-C), se puede evaluar y considerar el proceso de planificación de este proyecto como efectivo y adecuado a las necesidades del mismo, ya que al realizar las búsquedas de información previas al desarrollo, el consiguiente proceso de codificación pudo llevarse a cabo de manera controlada, eficiente y fluida, garantizando una solución integral, cuya estructura interna es comprensible y adecuada y donde su comportamiento corresponde al esperado según los requerimientos planteados.

IV. Diseño y estructura

En la figura 1 es posible observar un diagrama de arquitectura del sistema, inicialmente tanto el usuario de manera manual como el propio sistema operativo en sus protocolos de inicio, son capaces de ejecutar el programa implementado. El programa utiliza la librería `in-sensors` para consultar al procesador sobre la temperatura promedio de los núcleos existentes, para así proceder a modificar el registro incluyendo una nueva entrada, la cual contiene información como la fecha y hora de la medición, así como la temperatura registrada durante la misma. Una vez hecho esto, el programa evalúa la medición obtenida y, en caso de superar los $80C$, escribe en el activador de notificaciones un 1, o en su defecto un 0. Este archivo es leído constantemente por el programa notificador, de manera que, en caso de detectar un 1 en el contenido del activador, envía una notificación de sistema advirtiendo sobre la temperatura del procesador, o de lo contrario, no hace nada y procede a volver a leer el archivo.

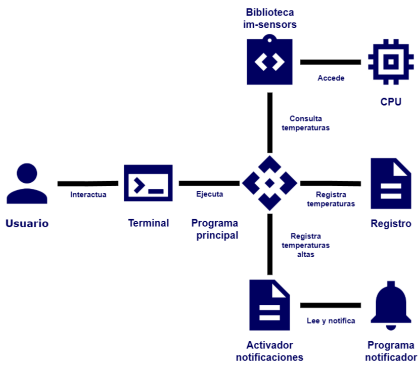

Figura 1: Diagrama de arquitectura del sistema

V. Instrucciones de uso

Inicialmente es necesario compilar los archivos programables (`.py`), para ello es necesario ejecutar los comandos:

```
chmod +x temperatureDaemon.py
chmod +x highTemperatureNotifier.py
```

Para interactuar con el programa central es posible utilizar el siguiente conjunto de instrucciones:

```
sudo ./temperatureDaemon.py start
sudo ./temperatureDaemon.py stop
sudo ./temperatureDaemon.py restart
```

Por su parte, el comando `start` es el responsable de poner a funcionar el programa central, de modo que este inicie la medición el registro de temperaturas. El comando `stop` detiene cualquier proceso de toma de datos existente y elimina el proceso del programa central y, finalmente el comando `restart` lleva a cabo los dos comandos anteriores de manera ordenada y consecutiva, iniciando con el comando `stop` y continuando con el `start`, de modo que el programa se detenga y vuelva a comenzar su funcionamiento desde cero.

Es necesario mencionar que, pese a que el programa central es capaz de funcionar por sí mismo y realizar las mediciones correspondientes y registrar entradas en el documento dedicado para dichos registros, este no es capaz de enviar las notificaciones de escritorio por su cuenta, por lo tanto, para ejecutar el programa que envía las notificaciones de escritorio es necesario ejecutar el siguiente comando:

```
nohup python3 highTemperatureNotifier.py &
```

De este modo, el programa se encontrará funcionando en su totalidad, garantizando mediciones acertadas y enviando notificaciones de escritorio cada vez que la temperatura del CPU supera los $80C$.

Adicionalmente, se incluye un programa llamado `stressDummy.py`, el cual hace uso de todos los núcleos del sistema para realizar operaciones aritméticas infinitas, de modo que exige rendimiento al procesador y aumenta intencionalmente la temperatura del sistema (Pese a que la temperatura del sistema aumenta gradual y lentamente, es recomendable ejecutar este programa con responsabilidad). Para su compilación, ejecución y detención se establecen los comandos que se listan a continuación:

```
chmod +x stressDummy.py
./stressDummy.py
Ctrl + C
```

Es necesario destacar que, una vez que son compilados por primera vez, los programas `temperatureDaemo` y `highTemperatureNotifier.py` se encuentran programados para ejecutarse al iniciar el equipo, por lo que el procedimiento de compilación detallado en este bloque de instrucciones de uso únicamente debe realizarse una vez, posteriormente los programas se ejecutarán automáticamente como parte de los procesos de inicio del sistema operativo.

VI. Resultados

La figura 2 demuestra una ejecución de los posibles comandos que soporta el programa central, los cuales son capaces de iniciar, detener y reiniciar el programa principal.

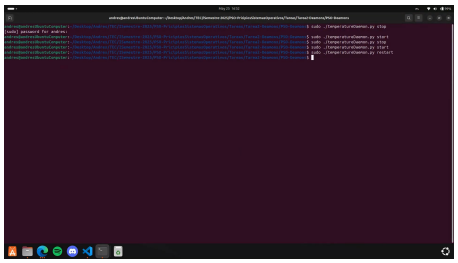

Figura 2: Ejecución de comandos del programa central.

La evidencia de la figura 3, ejemplifica el contenido habitual del registro de temperaturas, el cual recopila la información temporal de cada muestra y la temperatura registrada durante dicha medición. Este proceso se lleva a cabo cada $5s$.

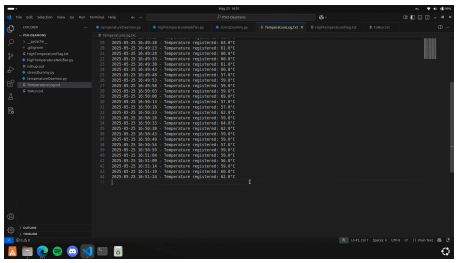

Figura 3: Contenido del archivo de registro de temperaturas.

En la figura 4 se demuestra una ejecución del programa `stressDummy.py`, el cuál activa los ocho núcleos del sistema para la realización de operaciones aritméticas que aumentan la carga de trabajo del procesador y por consiguiente, su temperatura.

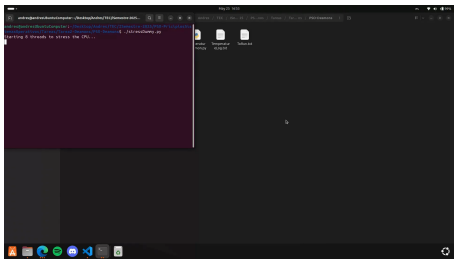

Figura 4: Ejecución del programa `stressDummy.py`.

Finalmente, en la figura 5 se observa una notificación de escritorio generada por el sistema debido a una alta temperatura, en el fondo de dicha figura es posible observar el registro de temperaturas, el cual evidencia la existencia de una temperatura de $89C$, el cual fue el responsable del posterior despliegue de la correspondiente notificación de escritorio.

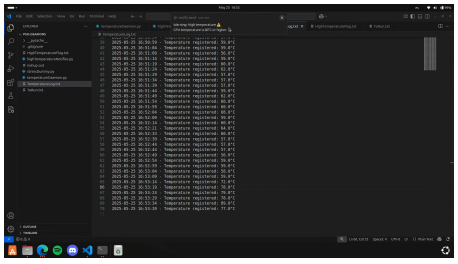

Figura 5: Notificación de escritorio debido a temperaturas superiores a $80C$.